**安装**

1、直接下载 / CDN

https://unpkg.com/vue-router/dist/vue-router.js

NPM

推荐使用淘宝镜像：

cnpm install vue-router

**简单实例**

Index.js

import Vue from 'vue/dist/vue.esm.js'

import router from './Router'             // 引入我们的路由文件

import App from './App'

var vm = new Vue({

  router,         // 应用路由

  el: "#container",

  template: '<App />',

  components: { App }

})

Router.js

import Vue from 'vue/dist/vue.esm.js'

import VueRouter from 'vue-router'

import About from './About'

Vue.use(VueRouter)

/\*定义路由 \*/

const routes = [

  { path: '/About', component: About }

]

// 实例化路由

const router = new VueRouter({

  mode: 'history',/\* 将默认的hash模式改变成history模式 \*/

  routes // （缩写）相当于 routes: routes

})

export default router;

App.vue

<template>

  <div id="app">

    <h3>App</h3>

    <!-- 跳转按钮 -->

    <router-link to="/">Go to Index</router-link>

    <router-link to="/About">Go to About</router-link>

    <!-- 用于放置匹配到的组件 -->

    <router-view></router-view>

  </div>

</template>

<script>

export default {

  name: "app",

  data() {

    return {};

  }

};

</script>

**<router-link>** 是一个组件，该组件用于设置一个导航链接，切换不同 HTML 内容。 **to** 属性为目标地址， 即要显示的内容。

**<router-link> 相关属性**

接下来我们可以了解下更多关于 <router-link> 的属性。

**to**

表示目标路由的链接。 当被点击后，内部会立刻把 to 的值传到 router.push()，所以这个值可以是一个字符串或者是描述目标位置的对象。

<router-link to="home">Home</router-link>

<!-- 使用 v-bind 的 JS 表达式 -->

<router-link v-bind:to="'home'">Home</router-link>

<!-- 不写 v-bind 也可以，就像绑定别的属性一样 -->

<router-link :to="'home'">Home</router-link>

<!-- 命名的路由 -->

<router-link :to="{ name: 'user', params: { userId: 123 }}">User</router-link>

<!-- 带查询参数，下面的结果为 /register?plan=private -->

<router-link :to="{ path: 'register', query: { plan: 'private' }}">Register</router-link>

**replace**

设置 replace 属性的话，当点击时，会调用 router.replace() 而不是 router.push()，导航后不会留下 history 记录。

<router-link :to="{ path: '/abc'}" replace></router-link>

**append**

设置 append 属性后，则在当前 (相对) 路径前添加基路径。例如，我们从 /a 导航到一个相对路径 b，如果没有配置 append，则路径为 /b，如果配了，则为 /a/b

<router-link :to="{ path: 'relative/path'}" append></router-link>

**tag**

有时候想要 <router-link> 渲染成某种标签，例如 <li>。 于是我们使用 tag prop 类指定何种标签，同样它还是会监听点击，触发导航。

<router-link to="/foo" tag="li">foo</router-link>

<!-- 渲染结果 -->

<li>foo</li>

**active-class**

设置 链接激活时使用的 CSS 类名。可以通过以下代码来替代。

<style>

.\_active{

background-color : red;

}

</style>

<p>

<router-link v-bind:to = "{ path: '/route1'}" active-class = "\_active">Router Link 1</router-link>

<router-link v-bind:to = "{ path: '/route2'}" tag = "span">Router Link 2</router-link>

</p>

注意这里 **class** 使用 **active\_class="\_active"**。

**exact-active-class**

配置当链接被精确匹配的时候应该激活的 class。可以通过以下代码来替代。

<p>

<router-link v-bind:to = "{ path: '/route1'}" exact-active-class = "\_active">Router Link 1</router-link>

<router-link v-bind:to = "{ path: '/route2'}" tag = "span">Router Link 2</router-link>

</p>

**event**

声明可以用来触发导航的事件。可以是一个字符串或是一个包含字符串的数组。

<router-link v-bind:to = "{ path: '/route1'}" event = "mouseover">Router Link 1</router-link>

以上代码设置了 event 为 mouseover ，及在鼠标移动到 Router Link 1 上时导航的 HTML 内容会发生改变。